**SetState Method:**

@protected

void setState(

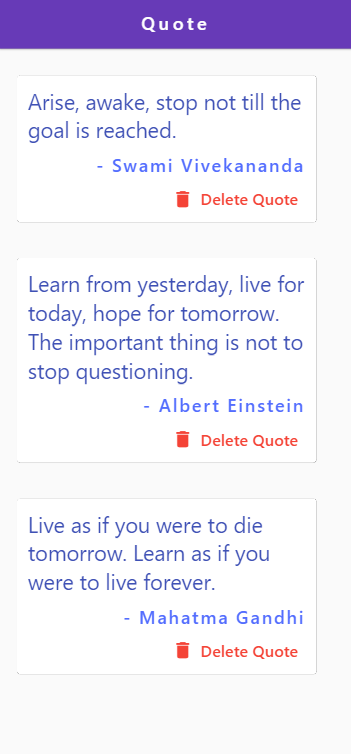
VoidCallback fn

)

* The provided callback is immediately called synchronously. It must not return a future (the callback cannot be async), since then it would be unclear when the state was actually being set.
* Calling setState notifies the framework that the internal state of this object has changed in a way that might impact the user interface in this subtree, which causes the framework to schedule a build for this State object.
* If you just change the state directly without calling setState, the framework might not schedule a build and the user interface for this subtree might not be updated to reflect the new state.

**TextButton Class:**

* A text button is a label child displayed on a (zero elevation) Material widget. The label's Text and Icon widgets are displayed in the style's ButtonStyle.foregroundColor. The button reacts to touches by filling with the style's ButtonStyle.backgroundColor.
* The text button's default style is defined by defaultStyleOf. The style of this text button can be overridden with its style parameter. The style of all text buttons in a subtree can be overridden with the TextButtonTheme and the style of all of the text buttons in an app can be overridden with the Theme's ThemeData.textButtonTheme property.
* The static styleFrom method is a convenient way to create a text button ButtonStyle from simple values.



**Main.dart :**

void main() => runApp(const MaterialApp(

debugShowCheckedModeBanner: false,

home: EchoList(),

));

class EchoList extends StatefulWidget {

const EchoList({super.key});

@override

State<EchoList> createState() => \_EchoListState();

}

class \_EchoListState extends State<EchoList> {

List<Quote> quotes = [

Quote(

text: "Arise, awake, stop not till the goal is reached.",

author: "Swami Vivekananda"),

Quote(

text:

"Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning.",

author: "Albert Einstein"),

Quote(

text:

"Live as if you were to die tomorrow. Learn as if you were to live forever.",

author: "Mahatma Gandhi"),

];

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.deepPurple,

title: const Text(

"Quote",

style: TextStyle(letterSpacing: 4),

),

centerTitle: true,

),

body: Column(

children: quotes

.map((quote) => QuoteCard(

quote: quote,

delete: () {

setState(() {

quotes.remove(quote);

});

}))

.toList(),

),

);

}

}

Tutorial 2 :

**Navigator Class:**

* Mobile apps typically reveal their contents via full-screen elements called "screens" or "pages". In Flutter these elements are called routes and they're managed by a Navigator widget. The navigator manages a stack of Route objects and provides two ways for managing the stack, the declarative API Navigator.pages or imperative API Navigator.push and Navigator.pop.

**PushNamed Method:**

@optionalTypeArgs

Future<T?> pushNamed<T extends Object?>(

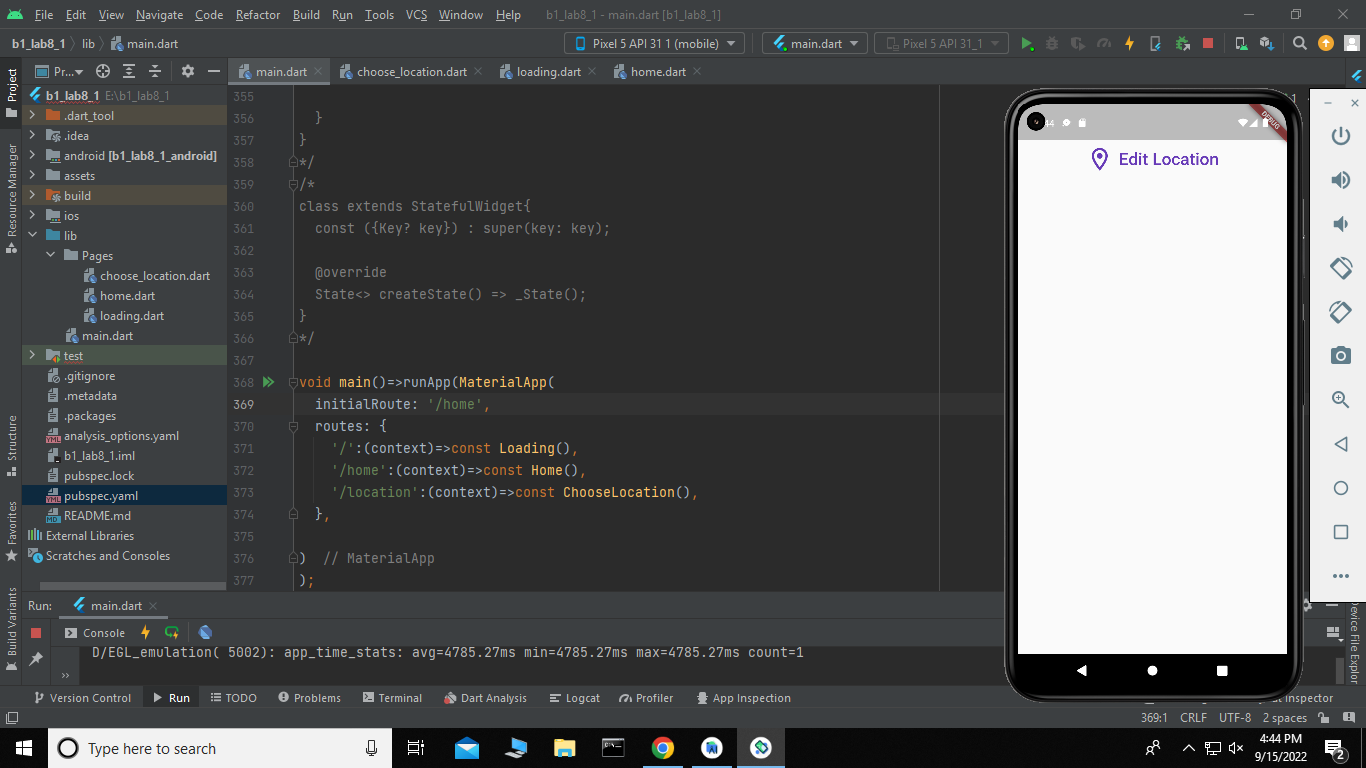
* BuildContext context,
* String routeName,
* {Object? arguments}

)

* Push a named route onto the navigator that most tightly encloses the given context.
* The route name will be passed to the Navigator.onGenerateRoute callback. The returned route will be pushed into the navigator.

**SafeArea Class:**

* A widget that insets its child by sufficient padding to avoid intrusions by the operating system.
* It will also indent the child by the amount necessary to avoid The Notch on the iPhone X, or other similar creative physical features of the display.
* When a minimum padding is specified, the greater of the minimum padding or the safe area padding will be applied.



By clicking the ‘Edit Location button ’ :

